How Well Can Existing Software-Support Processes Accomplish Sustainment of a Non-Developmental Item-Based Acquisition Strategy?

Graciano Nikolich

April 6, 2017

PUBLISHED BY
The Defense Acquisition University
Project Advisor: Dr. Craig Arndt
The Senior Service College Fellowship Program
Aberdeen Proving Ground, MD

Table of Contents

List of Figures

Abstract

The Department of Defense is increasingly moving toward software-intensive tactical systems. Software sustainment presents a differing set of characteristics over its hardware counterpart. To understand better how these differing characteristic may affect the current support processes established for the hardware-dominated landscape, this paper examines how a recent, ongoing, acquisition of a software-intensive tactical system (Joint Tactical Radio System) is aligning to the existing DoD and Army policy and guidelines for software sustainment. The paper further tries to identify potential disconnects presented by the DoD/Army's movement toward acquiring systems under the Non-Developmental Item (NDI) strategy. Under an NDI acquisition, the program manager acquires the end system with little to no development contribution or design insight. Recommendations are made to assist the Army in recognizing such challenges and considering modifications to the current processes.

Chapter 1 – Introduction

Background

In the Department of Defense (DoD) system acquisition process, sustainment[1] is the final phase of that cycle. Formal DoD instructions (DoDI) and Army regulations (AR) provide the guidance for a system to transition from development and fielding to sustainment. Furthermore, statutory law for core logistics capabilities mandates for no less than 50 percent of the system sustainment to be executed by a federal government organization (Title 10 United States Code (U.S.C.), sections 2464 and 2466). Historically concentration for this final phase of a product's life cycle has been aligned toward support of a hardware-centric design. That is not to say that software is not acknowledged within these instructions. For example, post production software support is a defined process that recognizes the presence of software within a system and the necessity to support it through the end of the system's life cycle. However, the guidance provided infers a system that has been developed through its acquisition milestones in joint participation with an Army program manager (PM) and the other party's development team (typically a prime contractor). That typically provides the PM with full design knowledge, from the initial allocation of system requirements (hardware vs. software), followed by prototyping, and culminating with the final design and production. Under this type of system development approach, the government's sustainment organization is well positioned to execute system sustainment by following a mature set of preparatory steps for the eventual transfer from the original equipment manufacturer to the government agent.

The trend over roughly the last decade by the Army has been from hardware- to software-centric tactical system designs. These designs introduce a unique set of sustainment variables that may not align well to the processes and procedures supporting hardware-centric system sustainment. To further drive down weapon acquisition costs, the DoD/Services are placing greater emphasis on leveraging commercially available software applications. An even further step in this strategy has been a shift to acquiring systems as a Non-Developmental Item (NDI). In the case of an NDI acquisition, a complete end-system is typically procured without a government funding contribution for its up-front development and maturation to production-level status. In effect, the government has no involvement with the up-front design of the system. An example of this acquisition strategy evolution is the Joint Tactical Radio System (JTRS) program. The majority of the JTRS functions, including some that in the past have typically been executed in hardware (e.g., cryptographic engines), are executed in software modules. This provides much greater flexibility for function and capability enhancements without having to redesign expensive hardware. A decision in 2014 by the Milestone Decision Authority identified the acquisition strategy for this program to be NDI based. In effect, the product would be purchased as an end item, government acceptance tested in developmental test and operational test environments, and then fielded. This is the first time a family of complex tactical (next-generation) communications systems is being procured in an off-the-shelf strategy. It is a reasonable assumption that these communications systems will experience frequent software design changes (based on what is witnessed in the commercial communications market). Thus the expectation is for perpetual changes driven by a variety of modifications (bugs, security vulnerabilities), increased functionality, adaptations to software operating systems, and even business models of companies (for example, the next version of software typically results in abandonment of previous versions, creating obsolescence dilemmas for customers).

Problem Statement

The trend to software-centric systems will not abate. Cost avoidance attempts, such as NDI acquisition strategies, will only increase. As this proliferation of software increases within the DoD weapon systems, the NDI pursuit will create hybrid software products (tactical systems) that will include elements of commercial off-the-shelf (COTS) software, government furnished software (GFS), open system software (OSS, a.k.a. freeware), and native system software, developed by a confluence of independent vendors. An NDI acquisition strategy does not afford the government team insight into system design (detail). Under these circumstance, how well can existing software-support processes accomplish sustainment of NDI-based acquisition strategy? How will a third-party government sustainment organization accomplish organic system readiness?

Purpose of This Study

The purpose of the study is to examine the impact that software-intensive systems, being acquired under the NDI strategy, will have on the existing support processes. The study examines the current set of policies, instructions, and regulations that guide system support, and assesses their sufficiency to accommodate military acquisition trends for procuring software-centric systems under an NDI construct. The review looks for potential disconnects between the guidelines and the actual situations this trend presents.

Significance of This Research

This study adds to the body of knowledge regarding the increase of software-centric tactical systems in the military and the challenges they present for life-cycle sustainment. The study concentrates on examining a communications family of radios. The study assumes an increase in NDI-type system acquisition as the DoD looks to leverage industry (commercial and/or defense) adoption of third-party software applications that avoid the need for original development. Under these assumed circumstance, it is important to determine how the existing support structure will accommodate this change, or whether changes in life-cycle sustainment approach are necessary to account for the dynamic situation presented by this software, and in particular by NDI system acquisition.

Overview of the Research Methodology

This is an evidence-based research project that uses case study methodology in order to examine the present-day acquisition policies, instructions, and regulations for life-cycle sustainment of systems, and asses their capability to sustain software- centric systems. The research focuses on the growing trend of the Army to acquire tactical systems under an NDI strategy. The research also reviews published reports and papers from academia, as well as a reviews Service-specific best practices for software development and support. The research includes perspectives taken from personnel from the Army's software support agencies (Communications-Electronics Command [CECOM]; Software Engineering Center [SEC]), as well as some defense industry players involved with delivering software-centric products for the DoD and Army.

Research Questions

Given the language of the current laws, policies, and regulations governing military system acquisition and life-cycle sustainment, how are the variables of software-centric products aligned to those of traditional hardware-centric products? How well can existing Army software-support organizations accomplish sustainment of NDI-acquired products?

Research Hypotheses

NDI acquisition is characterized by no research, development, test and evaluation funding provided by the government for the development of the system. The combination of NDI and software-based tactical systems is an attractive acquisition strategy because the former avoids the government cost of funding development, while the latter provides a much greater design flexibility for future additive functionality over the historically hardware-centric designs. This increased software content of a product will require a change in the policies and procedures governing life-cycle support of a traditional hardware-based system. Without considering the unique differences between hardware and software sustainment, the Army will face difficulties in maintaining the readiness of these tactical systems.

Objectives and Outcomes

The findings of this research will provide PMs with a better understanding of the characteristic differences between a hardware-centric versus a software-centric tactical system that is being acquired under an NDI strategy. Understanding these inherent differences will allow the PM to partner with the sustaining organizations by recognizing and accounting for such situations over the system's life cycle.

Limitations

This study is limited to the available statutes, policies, regulations, and professional articles available from institutions and journals. The proliferation of software into military systems is a relatively new situation in comparison to the overall existence of military weapons. The commercial industry, by its creation and publication of software standards (e.g., Object Management Group), has played a major role in bringing about the military Services' adoption of software-based capabilities for tactical systems. Much less has been experienced regarding sustainment of such systems. Most of the examined literature comes from academia. Limited Service experience is available to address the software-centric situation. The Services and DoD have published a handful of best practices, but the majority assume the government (PM) is part of the program's development phase, and thus a participant in the up-front knowledge of the software-based system being fielded. Much less information was found that reflected an NDI situation, which in effect acquires the system after it has been designed and built.

Chapter 2 – Literature Review

This chapter summarizes the literature review that was found regarding acquisition and life-cycle support of major weapon systems. The review includes academic publications about sustaining software-intensive systems in the DoD. The literature is organized to identify laws, policies, audits, and guidebooks governing the DoD and Army, and research papers and articles.

Laws, Policies, Audits, and Guidebooks

DoDI 5000.02 (DoD, 2017). This gives directions to PMs for major weapon system acquisition. It includes descriptions of a program’s acquisition milestones, required documentation, and planning for transition from development to the operation and support phase. The instruction addresses software development in Enclosure 3, system engineering (Section 11, Software), and system sustainment in Enclosure 6 (Life-Cycle Sustainment), including the development of a life-cycle sustainment plan that maintains affordable operational effectiveness of the system throughout its life cycle.

DoDI 8510.01(DoD, 2016). This describes the requirements for obtaining an authority to operate a fielded system. The emphasis of the document is on cybersecurity.

Title 10, United States Code. Section 2464 defines the necessity for the core logistics capability to be executed by a government-owned and government-operated entity in order to ensure a ready and controlled source of technical competence and resources for national defense. Section 2466 establishes that no more than 50 percent of the yearly depot-level maintenance funds may be used for contract support. These laws govern the role a sustainment organization must execute for a system transitioning form development/fielding to sustainment.

Memorandum on Intellectual Property and Software Optimization (Assistant Secretary of the Army for Acquisition, Logistics and Technology [ASA(ALT)], 2016). This identified the need for an intellectual property (IP) strategy as part of a program's acquisition strategy. The IP strategy is to address life-cycle support technical data needs (including documentation, rights, licenses, patents, copyrights, and trademarks).

***Defense Acquisition Guidebook* (*DAG*; Defense Acquisition University [DAU], 2013).** The *DAG* provides guidance to the PM regarding development and sustainment of software. The need for a strong application of software engineering principles is emphasized. The *DAG* advocates establishing a software team to address the acquisition strategy (i.e., what is to be developed, planned use of government off-the-shelf/COTS/OSS, mix/hybrid, etc.). Open system architecture is encouraged for ease of future sustainment and/or capability upgrades. The *DAG* provides recommendations for specifically addressing software in a software development plan, consideration for post-deployment software support (PDSS), a software data management approach, and consideration for data rights and software safety.

Section 4.3.18.4 describes the benefits and concerns of using COTS. For example, some of the benefits are that it reduces development time (none required), allows for faster insertion of technology, and lowers life-cycle costs by leveraging the commercial industrial base. Conversely, some of the concerns are embedment of proprietary functions, restricted rights, possible lack of access to design information, and difficulty in finding a suitable replacement once the vendor moves on to another application (marketplace drives COTS, not the government).

***United States Air Force Weapons Systems Software Management Guidebook* (United States Air Force, 2008).** This document provides guidance for organizations that acquire or sustain systems that involve significant development, integration or modification to the embedded software. An overview identifies activities necessary to have a successful system/software acquisition. It describes some common issues that historically drive software problems for PMs. Some examples include planning based on unrealistic expectations; software teams that are inadequately staffed, unstable, or incapable; and an ineffective systems engineering interface to the software development process.

Army Regulation 70-1 (Department of the Army, 2016a). Section 6-4 of the policy defines the requirements for software acquisition. It identifies the role the materiel developers play in software acquisition and defines what software generational languages are to be used. It indicates the need for Software Engineering Institute Level 3 compliance for a contractor's software development capability and process maturity.

Section 7-15 of the policy identifies the materiel developer's responsibility for a post-production support plan until such time that the materiel developer determines it is appropriate to transition the responsibility to the sustaining command. It states the system will not transition before the first full fiscal year after close of the hardware production line. The term "post-production software support" (PPSS) is applicable to systems that have transitioned to sustainment and the depot maintenance OP-29 process.

Army Regulation 700-127 (Department of the Army, 2016c). Section 8-10 identifies the materiel developer's responsibility for prudent software support planning for transition to sustainment. Note that software is an integral component of that materiel.

Reliability Tools (Army Materiel Systems Analysis Activity [AMSAA], 2016). AMSAA offers the PM two software tools to be applied in contracts that anticipate having a software component as part of the materiel solution. Software reliability is an important contributing factor in lowering the cost of system sustainment. The tools help the PM gauge the degree to which the system is meeting its reliability requirements, as well as provide guidance in establishing a reliability growth program. Recommendations are offered for establishing a software reliability program, as a subset of the overall system reliability program. A scorecard tool provides 57 elements to use in assessing the developer's software experience.

Executive Order 062-17 in Support of Software Solarium II (Department of the Army, 2016b). This order directs a follow-up discussion of software development and sustainment in the Army's portfolio of systems (Solarium I and II were conducted in September 2016 and February 2017, respectively). It identifies the problem as follows: "There is a lack of unity of effort in the development, testing, and sustainment of software in order to enable current and future Army warfighter functions in the execution of unified land operations" (p. 1.B). In order to address the problem, the order identifies four software Lines of Effort to be further discussed at the follow-up forum.

Lifecycle Sustainment Strategies for Acquisitions of Items Developed Exclusively at Private Expense; Suggested Considerations (Gomes, 2017). This guidebook offers the audience assistance in the type of language to be inserted into a solicitation that is acquiring items developed by vendors with their own funds. It addresses approaches to technical data packages, data rights, and other life-cycle considerations for sustainment strategies. It also offers a few examples of systems acquired, and provides samples of language for sections L and M within a request for proposal.

Software Communications Architecture (SCA), Version 4.1 (Joint Tactical Networking Center, 2015). "This architecture was developed to assist in the development of Software Defined Radio (SDR) communication systems, capturing the benefits of recent technology advances which are expected to greatly enhance interoperability of communication systems and reduce development and deployment costs" (p. x). The SCA intent is to "[provide] portability of applications software between different communications systems, leverage commercial standards to reduce development cost, reduce software development time through the ability to reuse design modules, and build on evolving commercial frameworks and architectures" (p. x). The SCA models the approach being promoted by tenets of Open Systems Architecture: in other words, to ease function upgrades and reduce sustainment cost through modular designs. The SDRs are configured to operate an application referred to as a waveform. A waveform is the set of transformations applied to information that is transmitted over the air and the corresponding set of transformations to convert received signals back to their information content. A Mobile Ad-hoc Networking waveform is a dynamic *ad hoc* network that continuously self-forms (and re-forms broken connections between its network members) without the need for any fixed infrastructure, such as cell towers.

National Defense Authorization Act for Fiscal Year 2010, Section 804 (2009). This section requires the DoD to develop and implement a new acquisition process for information technology systems.

Common Operating Environment Implementation Plan Core (ASA[ALT], 2011). This document provides direction for implementing the common operating environment (COE) to ASA(ALT)'s portfolio of systems.

Research Papers and Articles

Sustaining Software-Intensive Systems (Lapham & Woody, 2006). This technical note discusses the challenges of sustaining DoD systems that are increasingly software dependent. The note poses a series of questions that are then explored by the authors. The authors address use of COTS and highlight the specific challenges that acquisition presents.

Recommendations are made to the materiel-developing PM, along with the top 10 issues that should be addressed.

A Decision Framework for Selecting Licensing Rights for Noncommercial Computer Software in the DoD Environment (Gross, 2011). The report highlights the importance of a PM determining the correct software licensing strategy. It offers a framework to help determine the type of noncommercial software license a PM should pursue in support of the program. The various rights identified in the Defense Federal Acquisition Regulations (Unlimited, Government Purpose, Restricted, and Specifically Negotiated) and their variables are defined. Four questions are identified that should be asked in order to determine the program's licensing needs.

An Investment Model for Software Sustainment (Ferguson, 2013). This blog post offers some insight into the increasing cost of software sustainment (as much as 70 percent of total life-cycle cost for software). It discusses the Software Engineering Institute's (SEI's) development of a systems dynamic model that can be applied to a software product in order to forecast better a future event that will create a tipping point for the PM, thus giving the PM time to take action in advance.

Software Sustainment Now and Future (Lapham, 2014). This report discusses the criteria necessary to prepare a product to enter sustainment. It also looks at some future trends in software sustainment.

Modeling Software Sustainment (Ferguson, Phillips, & Sheard, 2014). This report describes a systems dynamic model that can be used by PMs to anticipate major product tipping point. A tipping point is defined as a sudden and dramatic change in the condition of the program. The model can be used to measure (1) operational performance, (2) operational needs analysis, (3) engineering and delivery, (4) capacity and capability, and (5) improvement funding. Feeder data will let the PM determine the appropriate course of action to get ahead of the situation.

Addressing Software Challenges for the DoD (McLendon, Scherlis, & Schmidt, 2014). This report characterizes the growth of software in the DoD's weapon systems and the commensurate cost of sustainment. The report further characterizes some distinctions between hardware and software (e.g., software does not wear out). The report concludes with a recommendation for a more holistic approach to software sustainment that addresses the technical, management, and business perspectives in a balanced manner.

Summary

A substantial amount of information was drawn from the statutory laws, regulations, and guidebooks that address system acquisition, including accountability for software development and sustainment. Based on the cursory review of these documents, little is said regarding NDI acquisition of capability. It is a category of acquisition characterized by conscious acceptance of limited design knowledge. The academic literature was a rich source of information largely cautioning about the challenges of NDI software sustainment. These resources highlighted the need for appropriate support planning during the development phases of programs. Nevertheless, that may not be entirely possible for an NDI acquisition.

There appears to be gaps between the government laws, policies, instructions, regulations, academic reports, and the conditions an NDI acquisition creates. The NDI will have a hybrid of originally developed software, special-purpose software (e.g., near real-time operating systems), COTS software, GFS (waveform applications), and OSS.

Chapter 3 – Research Methodology

Research Hypothesis

For this research project, the null hypothesis (H_0) is that the current set of statutes, policies, and regulations provide sufficient guidance for a PM, acquiring a system under an NDI strategy, to accomplish software sustainment throughout the life cycle. The alternative hypothesis (H_1) is that although the guidance and direction exist, an NDI acquisition presents the PM with a unique set of variables that differ from an engineering-and-manufacturing-development-phased acquisition, which requires a change to the governance process. For an NDI situation, a tailored support strategy may be required that combines the sustaining organization's core depot repair responsibility for hardware with the PM's continued sustainment responsibility for software under PDSS for the entire life-cycle (versus transitioning to PPSS). This has the added advantages of allowing for continued capability additions by the PM in support of the user community and of accommodating advancing software applications by the vendors.

Research Design

My approach to this project was to conduct evidence-based research. A case study methodology was used in order to examine the present-day acquisition policies, instructions, and regulations for life-cycle sustainment of systems, and assess their capability to sustain software-centric systems acquired under an NDI strategy. Focused interviews were conducted with members from Army Material Command (AMC), CECOM, SEC, and individuals in military-sector industry in order to gather their perspectives. Information was also gathered from various academic and software affiliated institutions. Technical articles were reviewed regarding management of software development efforts.

Bias and Error

The selection of the research topic was driven by my experience and continued involvement with the JTRS program. This experience may create a bias toward the belief that the current (hardware-oriented) AMC/CECOM support process will not be able to sustain a complex software system. JTRS efforts identified the dynamic nature of the software development environment. Through my participation in the development phase, concerns were raised that a complex software system was not going to fit the support framework of a hardware-based system. To mitigate my bias, I attempted to conduct interviews with other professionals in the field of acquisition. It is also possible that their perspectives introduce some bias based on past experience.

Chapter 4 – Findings

Collected Data

The objective of this research is to find out whether the existing software support processes can satisfactorily sustain products acquired under the NDI approach. Or are there structural changes necessary to the governance process for such acquisition strategies. The literature research indicated there is a general DoD awareness of the exponential growth in software within the military IT and tactical systems. Literature indicated a growing statutory and regulatory policy emphasis on software acquisition and sustainment complexities. This chapter will highlight a few major topics of concern to the PM that may be exacerbated by the increased migration to software-centric systems. Some of discussed topics are directly correlated to software sustainment challenges, while others topics (such as security vulnerabilities) have an indirect sustainment effect that may trigger a necessary change to the software design (i.e., code changes).

Life-cycle sustainment is described as follows: It translates force provider capability and performance requirements into tailored product support to achieve specified and evolving life-cycle product support availability, reliability, and affordability parameters. Life-cycle sustainment considerations include supply; maintenance; transportation; sustainment engineering; data management; configuration management; human systems integration; environment, safety (including explosives), and occupational health; protection of critical program information and anti-tamper provisions; supportability; and interoperability. Initially begun during the Materiel Solution Analysis (MSA) phase and matured during the Technology Maturation and Risk Reduction phase, life-cycle sustainment planning spans a system's entire life cycle from MSA phase to disposal. Maintenance (as a component of sustainment) is described as “Action necessary to retain or restore an item to a specified condition” (DAU, 2016). Software maintenance is further defined by the International Standard 14764 as including corrective maintenance, adaptive maintenance, perfective maintenance, and preventive maintenance (International Organization for Standardization/International Electrotechnical Commission [ISO/IEC], 1999, p. 6). Each term in effect categorizes the maintenance such that it can assist the owner in determining the immediacy of needed repair. These definitions are helpful in categorizing the subtle sustainment differences between hardware and software that are addressed in this paper. But regardless of the category, the use of these collective terms recognizes that software does not remain unchanged over its life cycle.

Current legislative policy in 10 U.S.C. 2464 states:

> It is essential for the national defense that the Department of Defense maintain a core logistics capability that is Government-owned and Government-operated (including Government personnel and Government-owned and Government-operated equipment and facilities) to ensure a ready and controlled source of technical competence and resources necessary to ensure effective and timely response to a mobilization, national defense contingency situations, and other emergency requirements. (p. 1447)

It is further stated in 10 USC 2466 that

> Not more than 50 percent of the funds made available in a fiscal year to a military department or a Defense Agency for depot-level maintenance and repair workload may be used to contract for the performance by non-Federal Government personnel of such workload for the military department or the Defense Agency. (p. 1449)

These statutory requirements present a challenge for large software-based systems, and even more so when these systems are acquired as NDI.

Analysis

There are a number characteristics associated with software-centric products that differ from the more commonly sustained hardware products. Guidelines from DoD and the Services offer the PM help in tackling software sustainment. However these guidelines largely assume the PM has started on the ground floor with the developer and possesses all the artifacts and knowledge needed for the system's transition to sustainment. This unfortunately is far removed from the situation presented by an NDI strategy. The following will identify how these guidelines align to the characteristics presented with JTRS, a highly software-centric product line, and the NDI acquisition strategy. The JTRS program is used as the case study for this research paper.

Stable Software Baseline

The general guideline is that a software baseline should be stable before it is transitioned to sustainment. A typical stability measure might be that no Category 1 (catastrophic) or 2 (critical) software trouble reports exist against the system. A sustainment transition plan, developed between the collective development organizations and the sustaining organizations (Lapham & Woody, 2006), provides the means for both parties to verify that stability measures have been met and the product has the necessary documentation to accomplish sustainment. To that end, as the software component of a system is developed, produced, and fielded, it transitions from PDSS to PPSS. The assumption is that once the system has completed fielding, its software has achieved stability and is now mature enough to be sustained by a maintenance PPSS organization. Software stability is very much correlated to software reliability. Software reliability is the measure translated from the system's required operational availability figure identified in a capability requirements document. In order for the software to be considered reliable (i.e., does not suffer from code defects, or “bugs”), it must execute the intended functions in the required manner. It must be repeatable, failure-free, consistent, and so on. In other words, it must be stable. As an acquisition assistance tool, AMSAA (2016) has developed software reliability language and a software reliability scorecard that can assist the acquiring PM in vetting the maturity of a system’s software. The tool can be effective at all stages of the program, but is most effective when applied at the earliest stages of development.

Under an NDI acquisition, it is up to the vendor to select any and all software that will run in the system (in this case a radio). Experience with JTRS has shown that the mix of software modules will be a hybrid of vendor-contracted software (government funding), vendor-native software (own funding), second-party tailored software (own funding; e.g., programmable cryptographic engines), and aggregates of COTS and OSS components. Each one will be governed by its originator’s intellectual property rules. Adding to this mix are the waveform (WF) applications, which are GFS. The vendor is required to run the WFs in the radio. Lastly, the radio (i.e., the WFs) must be managed on the battlefield by a government-developed network manager application. The configuration of this confluence of software will be jointly managed (vendor and government PM). From this mixed ensemble, you can quickly infer that finding a stable software baseline is challenging. Any one of these software modules may undergo a change (bug fix or vendors moving on to next version) at any point in time throughout its 20-year life cycle. In fact, the PM fully expects that an improved version of the WFs will be issued every 3 to 4 years. It is also acknowledged that an improved version of a WF will at some point break backward-interoperability with its predecessors. Similar conditions can be expected from COTS software vendors. Commercial vendors frequently improve their products and abandon support for previous versions.

At the beginning of chapter 4, I listed four categories of software maintenance (corrective, adaptive, perfective, and preventive). However, none of these captures yet another trigger for software maintenance: hardware obsolescence. In particular, hardware components have an effect on software. Obsolescence or supply chain interruptions of electronic components such as processors, field programmable gate arrays, and application-specific integrated circuits, will force the vendors to re-engineer the software to accommodate the next generations of these components. This collective dynamic environment presents the antithesis of a stable software baseline.

Documentation. A strong emphasis is placed on obtaining thorough (and complete) software design documentation to support sustainment. This is a reasonable expectation where the PM is contracting a vendor to develop a product. In such a situation, the PM can follow a standard systems engineering process that defines all desired software documents and conduct design reviews along the path from the initial requirements capture, requirements allocation to hardware or software. Figure 1 depicts a typical systems engineering process for product development. Following this process would generate the associated software documents at each step of the engineering iteration. Such documents would include the software development plan, design descriptions, requirements specifications, test documentation, interface requirement specifications, product specifications, and software version description.

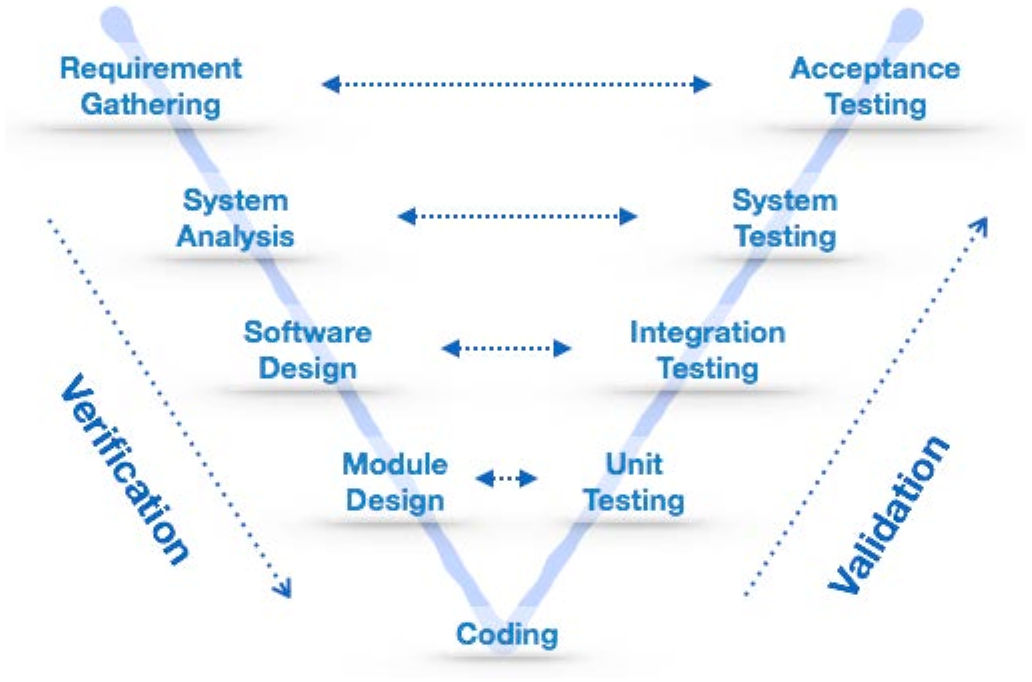

Figure 1 – Software Engineering V-Model
(Source: Tutorialspoint.com, 2017)

Through the contract, the PM's team is able to be involved in every step of the product design and to determine jointly when the software design portion is ready for the next step. Along the way, software design documents are generated to capture each step and development tools are identified. This also provides the PM with design insight for subsequent transfer of sustainment to a government owned/operated entity (per the 10 USC 2464/2466 requirements). However, the NDI acquisition strategy skips not only the development stage of the program, but also avoids providing any insight into how the product's software was engineered. In this situation, the PM is acquiring the product after it has been designed and built. The type and fidelity of the product's software documentation is unknowable. A contractual request can certainly be made for the desired set of document. However there may be many risks. The price may be high. The quality may be substandard (not governed by government based standards). The content may be incomplete (missing each step of the engineering process, or not written with a third-party user in mind). In some instances, the content may not be available due to proprietary claims. The prime vendor may not have access to all the software contained in the product. No assurance that an independent verification and validation was employed by the vendor. All these areas tend to drive up the cost of the product, making it a safe assumption that vendors most likely do not have the level of documentation required (i.e., cost avoidance).

As described below in the Authority to Operate (ATO) section, the sustaining organization will be faced with pursuing re-certifications of a fielded system. An updated and correct set of cybersecurity design documentation becomes critical in the pursuit of any re-certifications.

COTS software. Many will tout that leveraging COTS for the Army's business and tactical systems is beneficial to delivery of capability. From the financial avoidance to increased flexibility for additive capability, COTS offers advantages that should be exploited. There are, however, some drawbacks to leveraging COTS. The *DAG* (DAU, 2013) and an academic report (Lapham & Woody, 2006) highlight some of the concerns that must be addressed in the sustainment strategy. To mention just a few, COTS is very cyclical in nature. A COTS vendor may target a very fluid customer base, frequently abandoning prior application in pursuit of new business. The pedigree of COTS may be suspect. Quality of code may be much lower than required for tactical military needs. Revising COTS is highly discouraged, as it creates versioned orphans that may not have sustainability. Licensing agreements can vary and be very restrictive. COTS can be open source, freeware, or proprietary (Gross, 2011). Subsequent versions of COTS can outpace the system's infrastructure or digital processing budgets (McLendon et al., 2014). This can lead to hardware design upgrades.

Licensing Rights. Obtaining the right type of rights (unlimited rights, government purpose rights, or restricted rights) necessary for future sustainment of software is a key decision a PM needs to make early in the program. As product development proceeds, the PM may have to adjust the licensing rights strategy to accommodate the mix of software in the system. In a developmental program, that decision can be determined based on a vendor's proposed approach to developing that product (e.g., government-funded software development provides the government unlimited rights, mixed-funding provides Government Purpose Rights). The Software Engineering Institute offers a decision framework for determining the type of licensing rights that ought to be pursued by a DoD customer (Gross, 2011). It offers a series of questions to be answered by the customer in order to purchase the right level of a license. The *DAG* (Section 4.1.3.1, DAU, 2013) states,

> It is not uncommon for weapon system acquisitions to contain a mix of Government-off-the-shelf (GOTS) software with complete technical data and software rights, other software items with restricted Government purpose rights, and software with virtually no rights other than the commercial license to use or access the software. (p. 17)

In the JTRS NDI acquisition strategy, this is exactly the situation in which we find ourselves. The design is complete. Thus greatly limiting what rights the PM can insist on. If the vendor has selected third-party proprietary software modules as part of the overall software design, it is unlikely that these parties will offer unlimited or government-purpose rights. A good example of this is the industry's privately funded development of programmable cryptographic modules. These modules enable the radio to operate with multiple cryptographic algorithms and keys by executing the vast majority of functions in software. They provide the capability to simultaneously communicate on multiple networks, at differing security levels. In relative terms, the technology of these modules is very new. Thus the vendors have not offered the government any data rights. Perhaps data rights (or specifically negotiated license rights) may be offered at some point in the future, once the vendors no longer anticipate a business value for the software. However, it should be cautioned that providing rights does not provide needed software documentation, which may or may not exist in any useable form.

Multiple Software Versions. An ideal situation for any fielded product is that only a single configuration is in existence at any one time. The number of different configurations is highlighted as one of the major drivers of sustainment costs. That situation is depicted in Figure 2, a typical hardware-centric acquisition program.

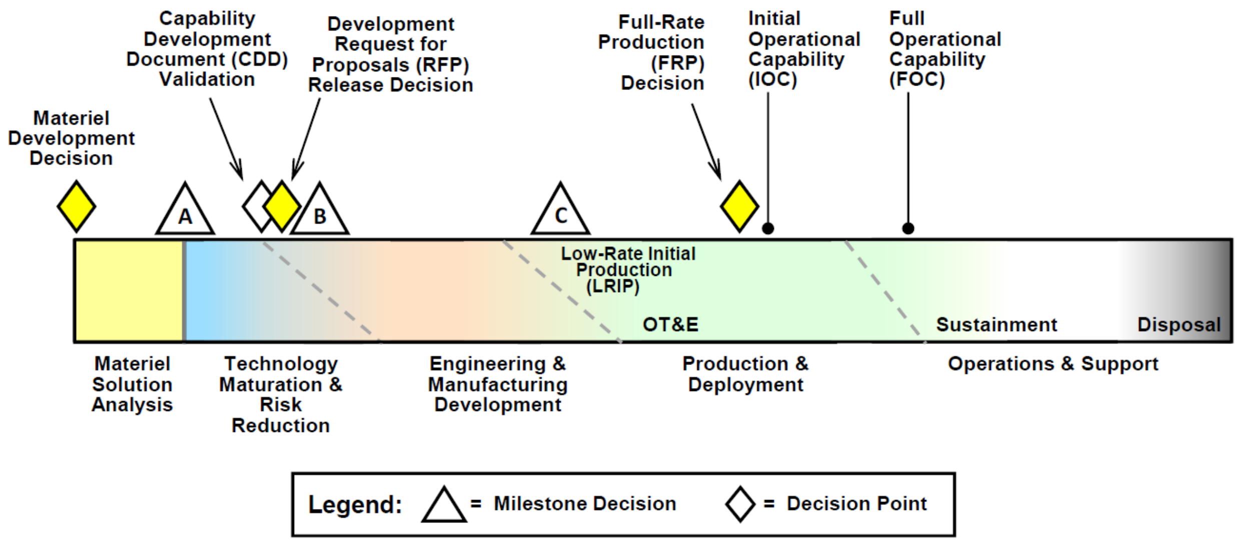
Capability Development Document (CDD) Validation
Development Request for Proposals (RFP) Release Decision
Full-Rate Production (FRP) Decision
Initial Operational Capability (IOC)
Full Operational Capability (FOC)
Materiel Development Decision
A
B
C
Low-Rate Initial Production (LRIP)
OT&E
Sustainment
Disposal
Materiel Solution Analysis
Technology Maturation & Risk Reduction
Engineering & Manufacturing Development
Production & Deployment
Operations & Support
Legend: = Milestone Decision = Decision Point

Figure 2 – Hardware Program
(Source: DoD, 2017, p. 11)

However, the multiple vendor mix of software contained within the radio products, and the computer-like electronic nature of the products themselves, will force multiple versions to simultaneously exist on the battlefield. Contributing to this problem is the Army's lengthy fielding timeline for the JTRS products (although certainly not unique to JTRS), which will stretch to more than 12 years in total. This creates a situation where initial fielded radios will be generationally behind the technology of their successors. Assuming a typical 2- to 3-year technology turnover, coupled with a 3- to 4-year period for waveform application capability upgrades, this will lead to multiple product versions coexisting in the field during the fielding and at the completion of fielding (roughly the 12-year mark). Therefore the JTRS situation is much more realistically depicted in Figure 3, a hybrid program acquisition that is software-centric. Configuration management will be critical in maintaining interoperability between JTRS radio form-factors and even own self-versions (within a form-factor). Periodic upgrades will have to be planned by the configuration control board.

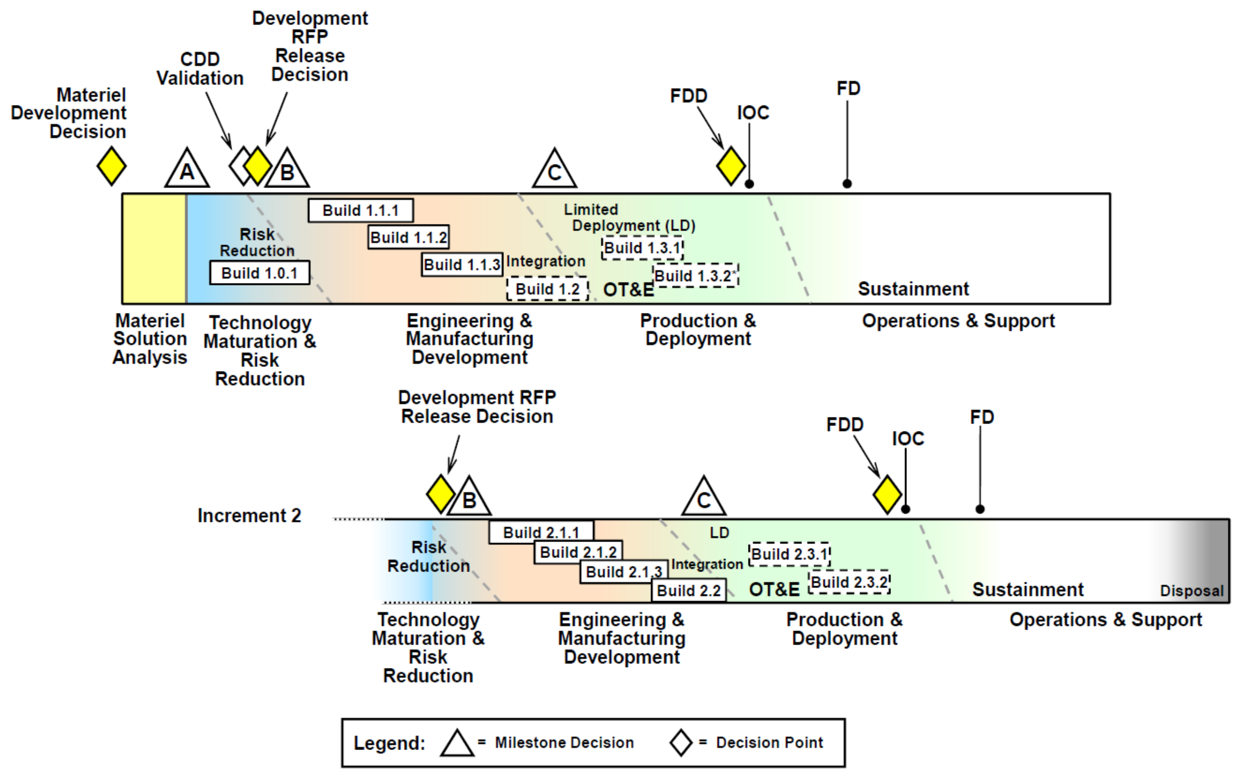

Figure 3 – Hybrid Program (Software Dominant)
(Source: DoD, 2017, p. 17)

Component obsolescence. It is a reasonable assumption that any system that remains in the field in excess of 5 years will face component obsolescence. In fact a number of Army regulations require the program manager to address the potential situation for component obsolescence in the acquisition strategy and life-cycle support plans (as a minimum). An additional requirement is to address diminishing manufacturing sources and material shortages. The typical approaches to resolving these areas is to acquire the technical data from the designing or manufacturing vendor(s) such that system components can be replaced by either re-engineering of the component or alternate sources identified. The PM is expected to address hardware and software within the system. However when one examines the construction of hardware versus the software in a system, the behavior of software is such that it has no logical sub-systems that can be replaced in the same manner as hardware sub-systems. Software tends to operate in a very tight functional dependency. Even a minor code change in one software component can have a large performance impact throughout the system. With this highly dependent framework of software components within a system, it is practically impossible to attempt positioning for software obsolescence of any one component without the very real potential of breaking the performance of the system.

One principle worth highlighting is the DoD’s recent emphasis on modular open systems approach (MOSA) for future competitive upgrades of sub-systems (DoD, 2017). This effort will also help the software obsolescence situation by creating logical functionality partitions with defined interfaces. In effect, the pursuit of MOSA will modularize the system’s software for easier sub-system replacement or upgrade. This is much like what has been the situation on the hardware side of the designs.

Authority to operate. The cybersecurity Risk Management Framework (RMF) for DoD Information Technology (DoD, 2016) defines the requirements for a system intended to be fielded to obtain an ATO in order to operate. The ATO is received from the Service designated approving authority (DAA), and will typically be valid for a 3-year period. The ATO is issued after an extensive cybersecurity assessment is conducted on the system. From its origins in 1997 as DITSCAP (DoD Information Technology Security Certification and Accreditation Process) to its current version as RMF, the process continues to be very labor intensive and lengthy (the final phase can take 24- to more than 30 months). The process typically starts at the very outset of product development (i.e., system categorization at Milestone A), and continues throughout its milestones till system fielding. Detailed design information is required for the assessment. Very often, design corrections are necessary along the development path to account for discovered cybersecurity vulnerabilities. The National Security Agency (NSA) is a key player on the system's path to ATO. Their information assurance assessment is an input to the overarching cybersecurity assessment. The acquisition PM must work closely with NSA in order to provide the agency with necessary design documentation for review. The increase in software-centric systems has outpaced NSA's ability to keep up with accreditation requests. The NDI acquisition strategy has further exacerbated the backlog. This is due to the vendors' desire that their products obtain NSA's information security certificates prior to contract awards. This in turn lowers the product's risk of not obtaining an ATO from the Service DAA.

Based on the limited 3-year ATO accreditation period, it is clear that a sustaining organization will have to pursue a system re-certification at least every 3 years (barring extensions). This period could be even shorter if a cybersecurity-critical design component undergoes a change. In case of a software-centric design, this could be triggered by a discovered vulnerability or a replacement of an obsolescent module that requires revetting for security posture. In such occurrences, the degree of design insight becomes critical in order for the owner of the product to convey the degree of change (via revised documentation).

Potential Actions to Improve Sustainment Posture

Compared to hardware, software's unique characteristics present different sustainment challenges. The following is a list of some potential actions that are already taking place or, if adopted, can assist in tackling these challenges.

Fundamental recognition of the challenge. It is axiomatic that in order to solve a problem, you first have to admit there is a problem. Evidenced by researched literature, the DoD and Services have recognized the inevitable growth of software-intensive systems in their military portfolios. A number of guidebooks are provided by DoD and the Services to assist the PM in acquiring and managing software systems. The Army has gone so far as to issue a set of executive orders, the latest being 062-17, in order "to provided key stakeholders the opportunity to illuminating the challenges associated with the exponential growth of software" (Department of the Army, 2016b, p. 2). The problem statement speaks to the "lack of unity of effort in the development, testing and sustainment of software" (Department of the Army, 2016b, p. 2). The goal for the invited organizations is to identify actionable lines of effort that will drive software life-cycle efficiencies.

Similarly, other agencies are recognizing the software acquisition sustainment challenges. The Army Contraction Command (Gomes, 2017) and agencies such as AMSAA (2016) are providing the PMs with acquisition strategy language and assessment tools to be used in requests for proposals that put the government sustainment organizations in improved positions to tackle software maintenance. In fact, the Army Contraction Command's suggested considerations document offers a case study based on two JTRS NDI programs, the Manpack and Rifleman-radio form-factors.

Academia is recognizing the challenges of software sustainment for DoD. SEI, in particular, has continued to highlight the increased use of software in fielded systems and has expressed concerns similar to those described in this paper's other sections. SEI is attempting to develop a systems dynamics model for use by DoD that would guide the decision to upgrade or replace software in the field (Ferguson et al., 2014. The model measures parameters such as threat, support technology, and workforce capacity to help quantify the cost versus value of a decision.

The efforts being made by the collective community are positive in that they are recognizing and tackling the challenges. However, the results of these efforts are still too new to assess their effectiveness.

Software architecture standards. Adoption of a standard software architecture is one method that can greatly promote design modularity. As highlighted earlier, the DoD is emphasizing MOSA for implementation into its future systems. If successful, the MOSA initiative should result in modular systems for both hardware and software. This should in turn create an opportunity for a sustainment organization to handle obsolescence at module level (rather than impacting the entire system). Academic research (McClendon et al., 2014) also points to the importance of system architecture in enabling effective sustainment. The research highlights the benefits for management (i.e., incremental capabilities delivered to the users) and to maintenance (i.e., replacement of individual modules versus entire suites). Better Buying Power 3.0, a DoD acquisition efficiency initiative, identifies MOSA as one of its key components under the “Incentivize Innovation in Industry and Government” principle (DoD, 2014).

The National Defense Authorization Act for Fiscal Year 2010 identified the need for a new acquisition process for information technology systems by DoD. In response to this task and the subsequent guidance from the Army CIO/G6, the ASA(ALT) published the COE implementation plan. The goal of COE is to “enable the Army to develop, test, certify and deploy software capabilities more quickly” (ASA[ALT]2011, p. iv). ASA(ALT), through its principal program executive offices, has applied the COE model to a portfolio of mission command systems. Program Executive Officer Command Control Communications–Tactical has adopted the COE model by pursuing a transition of its Command Control Communications and Computer systems from stove-piped to the common (shared) use of hardware and software components. The unique application of each system is retained while the common function modules are shared by all.

Similar to the principles of MOSA and COE, the JTRS has developed a software communications architecture (SCA; Joint Tactical Networking Center, 2015). This architecture is a specific standard for tactical radios that includes implementation of application program interfaces, promoting modularity through abstraction of general services (software or hardware components). The goal of SCA is to ease the integration of WF applications with the radio’s native software. The government’s WFs are provided to the vendors with a complete documentation package. The WFs are designed per the SCA definitions, thus creating a very modular set of code. In practice, any vendor that adheres to the SCA standards in their radio’s architecture design should be able to integrate subsequent versions of the WFs with minimal impact. Having the vendor adhere to the SCA promotes the general importance of creating a modular system (software) architecture for the overarching system design. As noted in *CrossTalk* (McLendon et al., 2014),

> Good architectural designs anticipate change by encapsulating variability to reduce cost and risk. In this approach, change-prone areas (such as hardware and communications infrastructures) are accessed via stable interfaces whose implementations can be replaced without undue side-effects on other software components. (p. 30)

The SCA therefore creates a software modular design that helps the sustainment organization make software sub-system changes much like the hardware sub-system counterpart.

Capability Maturity Model Integration (CMMI). Created by Carnegie Mellon University, "CMMI is a process level improvement training and appraisal program…required by many DoD and U.S. Government contracts, especially in software development. [It] can be used to guide process improvement across a project, division, or an entire organization" (Wikipedia, 2017). Figure 4 depicts the five levels for process maturity.

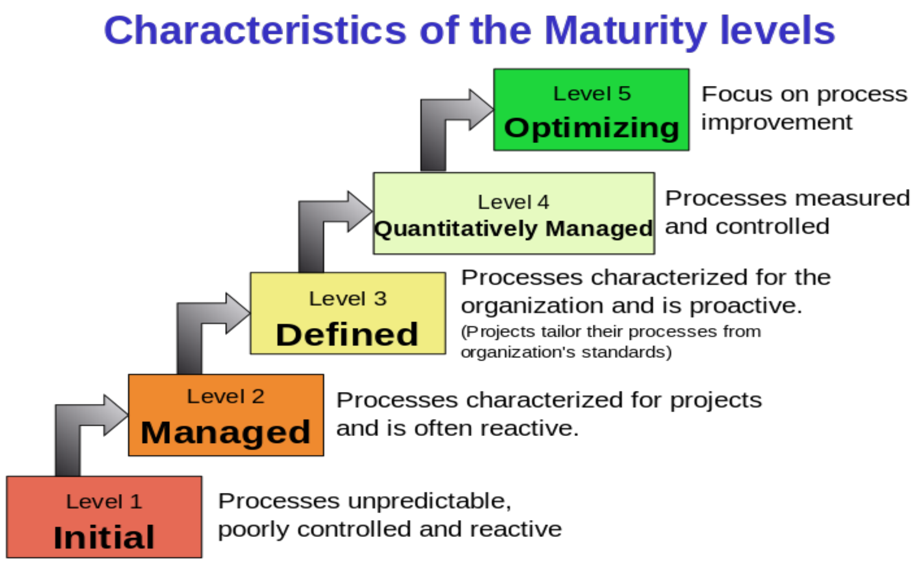
Characteristics of the Maturity levels
Level 5
Optimizing
Focus on process improvement
Level 4
Quantitatively Managed
Processes measured and controlled
Level 3
Defined
Processes characterized for the organization and is proactive.
(Projects tailor their processes from organization's standards)
Level 2
Managed
Processes characterized for projects and is often reactive.
Level 1
Initial
Processes unpredictable, poorly controlled and reactive

Figure 4 – Maturity Levels
(Source: Wikipedia, 2017)

The core goal of CMMI is to force discipline into the software development process. Companies that pursue excellence in software development look to be certified at the higher levels of performance. A minimum of a Level 3 certification has been the norm as part of DoD contracting requirements that involve large efforts in software (Department of the Army, 2016a, Section 6-4). It is noteworthy that level of certification is not a guarantee of program success. For example, a vendor can have a high-quality factory in place, but may still manufacture a faulty design. Nevertheless, the CMMI maturity-level assessment can be applied as an indicator for a disciplined approach to software development, thus increasing the potential for a successful code development.

Execute life-cycle support under PDSS. Per the Army acquisition policy (Department of the Army, 2016a, Section 7-15), the system transitions to sustainment after the close of the hardware production line. In the case of software, that is labeled as PPSS. For a radio portfolio such as JTRS, that can translate to decades of radio production. This is due to the very large quantity of product and the protracted integration into the Army's brigades. With a life cycle from 7 to 20 years (depending on the radio form factor), the production and fielding timeline alone overlaps the life cycle of the product itself. The lengthy fielding situation presents an opportunity for the PM to consider retaining the responsibility for the product's software sustainment until the end of the life cycle. In lieu of transitioning to PPSS (Operation & Maintenance, Army [OMA] funded), the system software continues to be sustained under PDSS (Other Procurement, Army [OPA] funded). Under the PDSS-funded sustainment, the PM has the latitude to continually add functional capability to the product in concert with the requirements community needs. As depicted in Figure 3, incremental capability can be fielded in a series of software builds. This continued total ownership of sustaining fielded radios gives the PM the flexibility to determine readiness of follow-on builds and allows for a coordinated capability integration into the Army's Training and Doctrine Command Concept of Operations.

Chapter 5 – Interpretation

Conclusions

The proposed H_0 of this research was that the current set of statutes, policies, and regulations provide sufficient guidance for a PM acquiring a system under an NDI strategy to accomplish software sustainment throughout the life cycle. The H_1 was that, although the guidance and direction exist, an NDI acquisition presents the PM with a unique set of variables that differ from an engineering and manufacturing development phased acquisition, requiring a change to the governance process. The conclusion reached by this author is that the basic system development and subsequent sustainment guidance are in a state of policy transition. To that end, H_1 appears to be more prevalent. Research indicated that growth of software in the military has not gone unnoticed by DoD and the Services. Published policies indicated the DoD has been increasingly focusing attention on software acquisition (related to business and tactical systems). In turn, the Army has also reflected these policies in their various regulations. However, these regulations have not been able to keep up with the pace and variety of situations being presented to the PM by industry. Arguably the most affected policies are those that are directed by Title 10 of the U.S. Code, pertaining to core logistics capability requirements of the federal government (specifically sections 2464 and 2466). A change may be necessary in order to recognize the exponential increase in software-driven tactical systems acquired not just under NDI, but in general terms. Whether intentionally or through commercial market forces, the Army—as the Service with the largest portfolio of systems—has recognized the functional and financial advantages that software-centric systems (non-NDI and/or NDI) provide over the more traditional hardware-based designs. Consequently, in addition to the various policies, the Army has started to address more directly the challenges with acquiring and sustaining software.

Evidence of that is the recently conducted Software Solariums (Sept. 2016 and Feb. 2017) that are bringing together every major Army acquisition and sustainment organization together to discuss software acquisition and sustainment challenges (Department of the Army, 2016b). It is a clear example of the Army's very real recognition of the need to coordinate across the entirety of the community. This is also an indication that, although a number of DoD and other Army policies and regulations speak to software acquisition and sustainment, the evolving real-world trends such as quick version turns on applications require continued reconsiderations of the acquisition approach.

The research indicated that DoD and the Services have identified a prudent set of policies and guidelines that address the way a PM should approach software acquisition for the situations where the government is part of the initial design phase (thus also most likely financing and dictating the necessary artifacts created in these early phases). Gap analysis reveals no substantive gaps in the set of guidelines outlining software design data that ought to be acquired. Of course, it is still incumbent on the PM to make the appropriate determination between data contents (including IP rights) and what the government will need to sustain the system. That determination should be reflected in the acquisition strategy document and the IP strategy within that document.

The guidelines are much less sufficient in dealing with situations where the PM is increasingly less involved with the system design, thus lacking insight into the particulars of the software design and/or its collective developers. The extreme situation in this regard is the NDI acquisition. The research indicated the DoD is poorly positioned to handle these type of situations. Proposed approaches to resolving such challenges still cling to the notions of obtaining necessary system design documentation (including software code) and the appropriate

IP rights for future sustainment by the government. That is not to say that acquiring the necessary artifacts (software documents, code, and development environments) does not have a place in the sustainment strategy. These items will be necessary in the event of vendor code abandonment or vendor departures. In such situations, the PM must have the ability to continue organic software sustainment, and for such anticipated situations the PM ought to establish an escrow account for the deposit of necessary design data.

The JTRS program that is used as the case study for this research presents both characteristics of the situation described above. The JTRS started as a DoD-funded development program that included the creation of all design documentation from capture of technical requirements, to their allocations to hardware or software, to the actual technical data packages for hardware and software. In addition, the communications applications (known as waveforms) that are now provided to industry as GFS were designed under this development phase. This largely describes the ideal situation of being on the ground floor of development and dictating what is to be delivered to the government. Now contrast this with the current JTRS program situation in which the Army is contracting only for the production of the NDI radios. Industry is provided the waveform applications as GFS. How to host (run) these waveform applications inside the radio is the vendor's decision. The only government oversight requirement is that the applications be capable of communication to other radios hosting the same waveforms. Thus you can see that the opportunity to select differing software architecture approaches is virtually endless. Note also that it is the Army's intent to continue upgrading these waveform applications for increased performance, adding to the challenge of continuously changing software. This is about as far away as one can get from the ideal situation of design-knowledge, maturity, singularity of fielded software, and software stability.

The research identified the movement toward a modular architecture approach to system design that increases the opportunity for function enhancements and/or future competition at the module level. These modularity principles are promoted through the DoDI 5000 series instructions regarding modular open systems approach, and via the Army's regulations in the pursuit of software COE. The COE breaks down stove-pipe Command, Control, Communications, Computers and Intelligence systems by using a common software operating system and treating the individual mission systems as applications that execute within this environment. Although not directly targeting sustainment, the migration toward modular software designs should have a beneficial impact on future software sustainment. For example, rather than a singular, monolithic, highly complex, integrated design, the modular design allows for much smaller scale, individual, module-code fixes or even replacements as a maintenance action.

Recommendations

It is a safe assumption that the DoD, and specifically the Army, will continue to be affected by the exponential growth in software-centric systems, with an ever greater opportunity to leverage NDI software. The JTRS program's NDI-based acquisition strategy is most likely the first of many such programs the Army will pursue in the future under the same type of strategy. By recognizing that software acquisition is a major challenge, the Army has taken the first critical step in developing a holistic software acquisition strategy including sustainment. It is recommended that this holistic life-cycle view consider technical and nontechnical perspectives. For example, forcing and rewarding modularity in system designs for the purpose of improved future additive functionality and competition is a technical perspective. This in turn enables the sustainment organization to assess a maintenance decision based on the cost of repairing or replacing at the software module level, which is more of a business perspective than a technical perspective.

It is recommended that the Army continue to monitor the JTRS radio program portfolio as a real-time case study regarding software sustainability approaches. The JTRS programs are a microcosm of all the variables that challenge the PM/Army in an NDI-type acquisition. The Army should learn from the JTRS program (positive outcomes as well as negative outcomes) and adjust acquisition strategies to accommodate the industry trends.

It is further recommended that the Army allow for greater PM flexibility in program decisions on transitioning to sustainment. It may be more advantageous to retain the program in the development/fielding phase throughout its life cycle, thus enabling the product to incrementally and continuously add capability. Note that a sustainment organization does not have the authority to increase capability of a fielded system. This tends to create disconnects in configuration management between versions of the system transitioned to sustainment and ones still under the PM purview. Long timelines (sometimes decades) for system fielding are especially vulnerable to this occurrence. The current financial structure—research, development, test and evaluation (RDT&E)/OPA for the development and production/fielding phase of a program, versus the Operational Maintenance, Army (OMA) for the sustainment phase—discourages the acquisition community from maintaining a product through its life cycle. That is because the acquisition community (i.e., the PM) is assessed for cost performance based on the established RDT&E/OPA program cost estimates. These are identified in the Acquisition Program Baseline (APB) document. Although the APB identifies the projected cost of sustaining the product, the OMA funds are not issued to the acquisition community. The incentive for the PM is to not breach the baselined RDT&E/OPA costs and to transition the product to the sustaining organizations as quickly as possible. An all-inclusive ownership of funds by the PM would reduce such incentives. It is a fair assumption that software-based products will provide a much greater opportunity to continuously add capability over the entire life cycle, thus retention of the system by the PM throughout the life cycle ought to be considered.

A final recommendation is that the Army consider new definitions in order to fully capture the varying types of software sustainment. For example, a criteria for transitioning to sustainment is that the software baseline has achieved stability. However software changes can be triggered by changes in electronic components such as obsolescence in processors or field-programmable gate arrays. It is likely that such changes will be perpetual and may label the system's software to be unstable. Such a known condition should not prevent the PM from transitioning the software to the sustainment organizations. Following or adopting a standard such as the one identified in the International Standard 14764 (ISO/IEC, 1999) may be prudent.

Limitations of the Study

The existing literature on the specific topic of an NDI acquisition in the Army tactical weapon portfolio was very limited. The JTRS program is the only one that was found to have been directed to acquire its systems under the NDI construct. Although it appears to be a rich case-study candidate, it offers a number of unknown situations that will be better answered in the future once we can gauge the program's future sustainment successes or failures. The limited time available for this research precluded an examination of the commercial industry approach to software sustainment. Exploring similarities in challenges and any resolution strategies may have been beneficial in considering them for adoption by the Army. In that regard, it is recommended that future research explore the habits and trends of the software-related industry in order to consider the applicability of their sustainment practices to the DoD/Army tactical systems.

References

Army Materiel Systems Analysis Activity. (2016). *Reliability tools*. Retrieved from https://www.amsaa.army.mil/CRG_Tools.html

Assistant Secretary of the Army for Acquisition, Logistics and Technology. (2011). *Common operating environment implementation plan core* (v3.0 draft). Retrieved from https://www.army.mil/e2/c/downloads/232001.pdf

Assistant Secretary of the Army for Acquisition, Logistics and Technology. (2016). *Intellectual property and software optimization* (Memorandum). Retrieved from https://acqdomain.army.mil/AcqBusiness/

Defense Acquisition University. (2013). *Defense acquisition guidebook*. Retrieved from https://www.dau.mil/tools/dag

Defense Acquisition University. (2016). *Glossary of defense acquisition acronyms and terms*. Retrieved from https://dap.dau.mil/glossary/Pages/Default.aspx

Department of the Army. (2016a). *Army acquisition policy* (Army Regulation 70-1). Retrieved from https://www.dau.mil/cop/rqmt/_layouts/15/WopiFrame.aspx?sourcedoc=/cop/rqmt/DAU%20Sponsored%20Documents/Army%20Acquisition%20Policy%20AR%2070%201.pdf&action=default&DefaultItemOpen=1

Department of the Army. (2016b). *Execution Order 062-17 in support of Software Solarium II*. UNCLASSIFIED FOUO (not available to the general public).

Department of the Army. (2016c). *Integrated product support* (Army Regulation 700-127). Retrieved from http://www.apd.army.mil/epubs/DR_pubs/DR_a/pdf/web/AR%20700-127_Web_FINAL.pdf

Department of Defense. (2014). *Better Buying Power 3.0*. Retrieved from http://bbp.dau.mil/docs/2_Better_Buying_Power_3_0(19_September_2014).pdf

Department of Defense. (2016). *Risk management framework (RMF) for DoD information technology (IT)* (DoDI 8510.01). Retrieved from http://www.dtic.mil/whs/directives/corres/pdf/851001_2014.pdf

Department of Defense. (2017). *Operation of the Defense Acquisition System* (DoDI 5000.02). Retrieved from http://www.dtic.mil/whs/directives/corres/pdf/500002_dodi_2015.pdf.

Ferguson, R. (2013, July 22). *An investment model for software sustainment* [Web log post]. Retrieved from https://insights.sei.cmu.edu/sei_blog/2013/07/an-investment-model-for-software-sustainment.html

Ferguson, R., Phillips, M., & Sheard, S. (2014, January/February). Modeling software sustainment. *CrossTalk*. Retrieved from http://static1.1.sqspcdn.com/static/f/702523/24156562/1388991346503/201401-Ferguson.pdf?token=zI2qLlueENWp49rwPcke1HPmDMQ%3D

Gomes, G. M. (2017). *Lifecycle sustainment strategies for acquisitions of items developed exclusively at private expense; Suggested considerations* (version 1.0.0, 2017JAN17). Aberdeen Proving Ground, MD: U.S. Army Contracting Command. Retrieved from https://www.fbo.gov/index?s=opportunity&mode=form&id=6bfaffad86ab2f4ca5ce7cecf4481185&tab=core&_cview=0

Gross, C. (2011). *A decision framework for selecting licensing rights for noncommercial computer software in the DoD environment* (Technical Report CMU/SEI-2011-TR-014; Pittsburgh, PA: Software Engineering Institute. Retrieved from http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9995

International Organization for Standardization/International Electrotechnical Commission. (1999). *International Standard 14764* (Information Technology–Software Maintenance). Retrieved from http://bcc.portal.gov.bd/sites/default/files/files/bcc.portal.gov.bd/page/adeaf3e5_cc55_4222_8767_f26bcaec3f70/ISO_IEC_14764.pdf

Joint Tactical Networking Center. (2015). *Software communications architecture (SCA), Version 4.1*. Retrieved from http://www.public.navy.mil/JTNC/SCA/Pages/sca1.aspx

Lapham, M. A. (2014, January/February). Software sustainment—Now and future. *CrossTalk*. Retrieved from http://static1.1.sqspcdn.com/static/f/702523/24156563/1388991346710/201401-Lapham.pdf?token=U2fffkSCplGsh2ajafdxf5NM2kE%3D

Lapham, M. A., and Woody, C. (2006). *Sustaining software-intensive systems* (Technical Note CMU/SEI-2006-TN-007). Pittsburgh, PA: Software Engineering Institute. Retrieved from http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=7865

McLendon, M., Scherlis, B., & Schmidt, D. C. (2014, January/February). Addressing software sustainment challenges for the DoD. *CrossTalk*. Retrieved from http://static1.1.sqspcdn.com/static/f/702523/24156564/1388991346767/201401-McLendon.pdf?token=%2FkdpaopBMsQirtEmJ8Gmvg3%2BnTg%3D

National Defense Authorization Act for Fiscal Year 2010, Public Law 111-84, Sec. 804, 10 U.S.C. 2225 note (2009). Retrieved from https://www.gpo.gov/fdsys/pkg/PLAW-111publ84/pdf/PLAW-111publ84.pdf

Title 10, United States Code, Ch. 146, Sections 2464–2466. Retrieved from http://uscode.house.gov/view.xhtml?req=granuleid%3AUSC-prelim-title10-chapter146&saved=%7CZ3JhbnVsZWlkOlVTQy1wcmVsaW0tdGl0bGUxMC1zZWN0aW9uMjQ2NA%3D%3D%7C%7C%7C0%7Cfalse%7Cprelim&edition=prelim

Tutorialspoint.com. (2017). *Software engineering—Quick guide*. Retrieved from http://tutorialspoint.com/software_engineering/software_engineering _quick_guide.htm

United States Air Force. (2008). *United States Air Force weapon systems software management guidebook* (Version 1, Abridged). Retrieved from http://www.acqnotes.com/ Attachments/USAF%20Weapon%20System%20Sofware%20Management%20Guide.pdf

Wikipedia. (2017). *Capability maturity model integration*. Retrieved from https://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration

Glossary of Acronyms and Terms

AMC	Army Material Command
AMSAA	Army Materiel Systems Analysis Activity
APB	Acquisition Program Baseline
ASA(ALT)	Assistant Secretary of the Army for Acquisition, Logistics and Technology
CECOM	Communications and Electronics Command
COE	common operating environment
COTS	commercial off-the-shelf
DAA	designated approving authority
DAG	Defense Acquisition Guidebook
DoD	Department of Defense
DoDI	Department of Defense Instructions
GFS	government furnished software
H_0	null hypothesis
H_1	alternate hypothesis
IP	intellectual property
JTRS	Joint Tactical Radios System
MOSA	Modular Open Systems Approach
MSA	Materiel Solution Analysis
NDI	Non-Developmental Item
NSA	National Security Agency
OMA	Operational Maintenance, Army
OPA	Other Procurement, Army

OSS	open source software
PDSS	post-deployment software support
PPSS	post-production software support
PM	project manager/program manager
RDT&E	research, development, test and evaluation
RMF	Risk Management Framework
SCA	software communications architecture
SDR	Software Defined Radio
SEI	Software Engineering Institute
WF	Waveform

Note

[1]The terms “maintenance” and “sustainment” are at times used interchangeably by the general community. While the term “software maintenance” does have a broadly accepted Institute of Electrical and Electronics Engineers definition as “The process of modifying a software or component after delivery to correct faults, improve performance or other attributes, or adopt to a changed environment” (Lapham & Woody, 2006, p. 1), the term “sustainment” does not have a generally accepted definition. Therefore this paper will use an SEI definition of sustainment as “The processes, procedures, people, material, and information required to support, maintain, and operate the software aspects of a system” (Lapham & Woody, 2006, p. 2). The Defense Acquisition Guide uses the general term of “operations and support” for a system that has completed production and been fully fielded (DAU, 2013).

Author Note

The author is an acquisition professional, with a Defense Acquisition Workforce Improvement Act Level III certification in Program Management and Systems Engineering. Correspondence concerning this paper should be addressed to Graciano.nikolich.civ@mail.mil.